プログラムの形式的正当性 (情報科学の数学的理論)

AUTHOR(S):

謝, 章文

CITATION:

謝, 章文. プログラムの形式的正当性 (情報科学の数学的理論). 数理解析研究所講究録 1972, 156: 167-182

ISSUE DATE:

1972-08

URL:

http://hdl.handle.net/2433/106853

数理解析研究所講究録
156巻 1972年 167-182

プログラムの形式的正当性

京大 工学部 謝 章文

§1. 序

文字列としてのプログラムを形式的に取り扱い、そのプログラムの構造および諸性質を考察する目的で、プログラミング言語、プログラミングおよびプログラムの形式的モデルをそれぞれ設定する。形式的取り扱いの基礎を与えるプログラミング言語は、その構文解析によって得られる幾つかの集合の集合として特徴づける。さらに、プロセデュアとしてのプログラムの一般的正当性すなわち全ての正しいプログラムに存在する共通の性質について考察し、与えられたプログラムのモデルについて、形式的に判定できる正当性およびそのアルゴリズムについて述べる。最後に、プログラムの一般的正当性と求められたプログラム形の形式的正当性の関係について述べる。

§2. プログラミング言語

ここでは、プログラムの構造を形式的に取り扱うために、必要な表現機能をそなえた形式的言語について考察する。

〔定義1〕　Nはn_iを要素とする可附番無限集合、D^kは有限ないし可附番無限集合とするとき（$k=1,2,\cdots,m$）、Nの全ての要素n_iに対して、ordered $(m+1)$-tuple $(n_i, D_i^1, D_i^2, \cdots, D_i^m)$, ただし $D_i^j \subseteq D^j$, $j=1,2,\cdots m$, が1つただ1つ与えられているとき、n_iを m-value name, D^jを m-value name の j-range という。また、$\#D_i^j = 1$[*1] のとき、n_iを j-constant, $\#D_i^j > 1$ のとき、n_iを j-variable, D_i^jを n_i の j-domain という。さらに、各 j に対して、m-value name n_i に1つただ1つの ordered pair (n_i, d) $d \in D_i^j$ が与えられることを、n_iが j-define されるといい、(n_i, d), d をそれぞれ n_i の j-name relation, j-denotation という。n_iが j-define されており、j-name relation (n_i, d) が与えられているとき、(n_i, d') $d' \in D_i^j \wedge d \neq d'$ でそれを置き換えることを、n_iを j-redefine するといい、また、n_iによって d が denote されることを、n_iが j-refer されるという。

〔定義2〕　2-value name の recursive set の 1, 2-range がそれぞれ finite, recursive set であるとき、2-value name を Identifier, 前者の要素を Address, 後者の要素を

*1　$\#D$ は集合 D の cardinal number を表わす。

Valueと呼ぶ。

現実のプログラミング言語では、1-constantすなわちdynamic registerなどは、implicitに表現されている。また、2-constantはliteralである。

〔定義3〕　constantを含まない1-value nameの集合のrangeがnameの集合であるとき、その集合の要素をmeta-variableと呼ぶ。

〔定義4〕　Statement Schema Form SSFを次のように定める。

$$SSF_i = \lambda v_i^{1d} v_i^{1r} v_i^{2d} v_i^{2r} \overline{SSF_i} \quad (*2)$$
$$\overline{SSF_i} = \langle F_i, v_i^{1d}, v_i^{1r}, v_i^{2d}, v_i^{2r} \rangle \qquad (1.1)$$

ただし、$v_i^{(1)(d)}_{(2)(r)}$：meta-variableの集合。右肩の添字はそのmeta-variableの位置に代入されるIdentifierの(1)(2)-denotationが(define / refer)されることを意味する。

F_i：$v_i^{1d} \cup v_i^{1r} \cup v_i^{2d} \cup v_i^{2r}$の要素のmeta-variableに代入されるIdentifiers上のfunction symbolの集合。

〔定義5〕　形式的プログラミング言語を次のように定める。

$$\mathbb{L} = \langle \mathbb{N}, \mathbb{D} \rangle \qquad (1.2)$$

ただし、$\mathbb{N}$；Identifierの集合。　$\mathbb{D}$；SSFの集合。

$\mathbb{N}$は4つの部分集合にわけられる。$\mathbb{N} = N_1 \cup N_2 \cup N_{C1} \cup N_{C2}$

$N_{1(2)}$；1(2)-domainが空集合でない1(2)-variableの集合。

*2. $\lambda v E$はE内の集合vの要素を全てboundすること。

$N_{C1(C2)}$: 1(2)-constantの集合。 ここでは、議論を簡単にするため、compiler levelの言語を考えることにする。したがって、 $N_1 \cap N_2 = \phi$ [*3] と仮定する。

〔定義6〕 Statement Schema SS は次のように定まる。

$$SS_i = \sum_{N_1' \cup N_{C1}',\ N_2'' \cup N_{C2}'}^{V_i^{1d} \cup V_i^{1r},\ V_i^{2d} \cup V_i^{2r}} SSF_i \Big| \quad *4$$

$$= \langle F_i, V_i^{1d}, V_i^{1r} \cup C_i^1, V_i^{2d}, V_i^{2r} \cup C_i^2 \rangle \qquad (1.3)$$

ただし、$N_1' \subset N_1$, $N_2' \subset N_2$, $N_{C1}' \subset N_{C1}$, $N_{C2}' \subset N_{C2}$, $C_i^{1(2)}$: 1(2)-constantの集合.

〔定義7〕 言語$\mathbb{L}$の任意のStatement Schemaの並びが与えられたとき、一意に各Statement Schema SS_j が定まり、かつ、$V_j^{1d}, V_j^{1r}, V_j^{2d}, V_j^{2r}, C_j^1, C_j^2$ と $\lambda V_j^{1d} V_j^{1r} V_j^{2d} V_j^{2r} \overline{SSF_j}$ が求まるとき、言語$\mathbb{L}$はUnambiguousであるという。

以後の議論において、プログラミング言語とはUnambiguousなものを指すことにする。

§3. プログラミングとプログラム

はじめに幾つかのtermを定める。集合$\mathbb{D}$の要素の重複順列をSSF-seqという。SSF-seqにおいて、各SSF_iごとにboundされているmeta-variableを、意味上の変更なしにSSF-seqの先頭でboundするようにした形をPrenex SSF-seqという。Prenex SSF-seqに出現するV^1, V^2のmeta-variableのgroupingをそれぞれ

*3 ϕ: 空集合。 *4 $S_N^V E|$: E内の集合Vの要素を、全べて、集合Nの要素で適当に代置したもの。

Label Linkage, Variable Linkage, 両者を Name Linkage という。Name Linkage に従って、Prenex SSF-seg の各 group の meta-variable を代表 meta-variable で置きかえたものを Seg-Form, Seg-Form に区切りマーク (END line) を追加したものを PS-Form という。PS-Form の V^1, V^2 の各 meta-variable へそれぞれ distinct な N_1, $N_2 \cup N_{c2}$ の要素を割り当てることを、Label Assignment, Variable Assignment, 両者を Name Assignment という。

〔定義 8〕　次の3つの Process を言語 $\mathbb{L}$ 上の形式的プログラミングと定義する。

1. $\mathbb{D}$ 上の (Prenex) SSF-seg の作成。　2. Prenex SSF-seg の Name Linkage.　3. PS-Form の Name Assignment.

〔定義 9〕　形式的言語 $\mathbb{L}$ 上の形式的プログラミングによって作成される Statement Schema の並びを形式的プログラム形という。

直観的な形式的プログラムの解釈は、"その指令に従って、計算機がなんらかの動作を連続して行なえるもの" である。Nondeterministic な Program, 無意味な部分、無意味な Action の指令を含む。ただし、次の仮定を設ける。

〔定義 10〕　そのプログラム内で全ての denotation (name relation) が定められる Identifier を "Inner Identifier" そうでないものを "External Identifier" という。

〔仮定〕　Program の External Identifier は既に外部でもって、その denotation が定められているとする。

この仮定より、External Identifier は計算機の動作可能性には影響を与えず、プログラムの制御可能性(flow および value)に影響する。

§4. プログラムの正当性の概念

この議論において、プログラムとは、停止することを意図するもののみに限定する。

〔定義11〕　プログラムの一般的正当性とは、全ての正しいプログラムに共有される性質とする。

〔定義12〕　プログラムの一般的正当性とは、それが、ある Algorithm または Semi-Algorithm の表現であることとする。

〔定義11〕は〔定義12〕で置きかえられるものと考える。しかし、現在 Algorithm のクラスの形式的定義およびそのクラスの要素による Algorithm の形式的定義は提案されているが、Algorithm そのものの直接的定義は与えられていない。従来、提案されている Algorithm の概念にとって本質的であると考えられる幾つかの一般的特徴は、1)段階性　2)決定性　3)要素性　4)方向性　5)無限性　等である。これに着目して次節以下でプログラムの形式的正当性を定義する。ここでは、一般的正当性について、次のような定義を採用する。

〔定義13〕　プログラムを P、その Input Domain を D_x、

Input Vector を ξ とするとき．$\forall \xi (\xi \in D_x)$ に対して，$P(\xi)$ の Computation が finite で かつ 常に同一の final result を得るならば．P は totally correct であるという。

〔定義 14〕　プログラムを P、その Input Domain D_x, Input Vector を ξ とするとき．$P(\xi)$ の Computation が finite で かつ 常に同一の final result を得るような ξ ($\xi \in D_x$) が存在するならば．P は partially correct であるという。

§5．プログラムの形式的正当性（その1）

5－1　L－Correctness

形式的言語 $\mathcal{L}$ 上の形式的プログラム形を次のように表わす。

$$PS = S_1 S_2 S_3 \cdots S_n E \qquad (1.4)$$

S_i：　Statement Schema の occurrence，簡単のため．その Statement Schema も表わすものとする。E：　区切りマーク．

$$\begin{aligned} S_i &= \langle F_i, V_i^{1d}, V_i^{1r} \cup C_i^1, V_i^{2d}, V_i^{2r} \cup C_i^2 \rangle \\ E &= \langle \#, Le \rangle .\quad \#.\ 終わりマーク.\ Le \in N_1 \end{aligned} \qquad (1.5)$$

〔定義 15〕　$i \neq j$ つ．$V_i^{1d} \cap V_j^{1d} = \phi$ ならば．PS は L－deterministic である。また．$(\bigcup_{i=1}^{n} V_i^{1r}) \cup \{Le\} \subseteq \bigcup_{i=1}^{n} V_i^{1d}$ ならば．PS は L－close である。　PS が L－deterministic かつ L－close ならば．L－correct であるという。

〔命題 1〕　与えられた PS が L－correct であるかどうかは決定可能である。

〔定義16〕　$(\bigcup_{i=1}^{n} V_i^{ld} - \bigcup_{i=1}^{n} V_i^{ir})$ の要素を r-label という。PS の全ての r-label を λ[*5] で置換したものを L-標準形という。

〔命題2〕　PS′ を PS の L-標準形とするとき、$PS \equiv^{*6} PS'$。

これ以後、PS とは L-correct, L-標準形のものを指すこととする。

5-2 G-Correctness

〔定義17〕　PS から次のプロセスによって作られる Digraph を GPS という。　$GPS = \langle \mathcal{N}, A \rangle$, $\mathcal{N}$: labeled node の finite set, A: arc の finite set。

1. 各 Statement Schema S_i に node n_i を対応づける。
2. $V_i^{ir} = \phi$ なる S_i の n_i から n_{i+1} へ arc を出す。
3. $V_i^{ir} \neq \phi$ なる S_i の n_i から、各 $L_j \in V_i^{ir}$ に対して、$L_j \in V_k^{ld}$ なる S_k の n_k へ arc を出す。
4. Start node n_0 を設け、n_0 から $Le \in V_i^{ld}$ なる S_i の n_i へ arc を出す。$Le = \lambda$ のときは n_1 へ arc を出す。

〔定義18〕　次の Induction Definition によって定まる node の集合を、それぞれ 到達可能集合 $\mathcal{N}_S$, 停止可能集合 $\mathcal{N}_h$ という。

1. Start node $n_0 \in \mathcal{N}_S$.
2. $n_i \in \mathcal{N}_S \wedge n_i n_j \in A$ ならば $n_j \in \mathcal{N}_S$.
3. 1, 2 によって定まるものだけが $\mathcal{N}_S$ の要素である。

1′. Halt node $n_i \in \mathcal{N}_h$.

*5. λ: 長さゼロの記号。　*6. 全ての Interpretation のもとで強等価であること。

2'. $n_j \in N_h \wedge n_k n_j \in A$ ならば $n_k \in N_h$.

3'. 1', 2'によって定まるものだけがN_hの要素である.

ただし Halt node とは 言語$\mathcal{L}$によって定まる Halt Statement S_i に対応する node のことである。

〔命題3〕 PSが Halt Statement を含まないとき. $N_h = \phi$.

〔定義19〕 PSのGPSにおいて. $N_S \supseteq N_h \wedge N_h \neq \phi$ ならば PS は G-correct, とくに $N_S \neq N_h$ のとき Weakly G-correct. $N_S = N_h$ のとき Strongly G-correct という。

〔命題4〕 PSが G-correct であるかどうか決定可能である。

〔定義20〕 $N - N_S$ の要素を r-node, 対応する S_i を r-Statement という。PSの全ての r-Statement を入で置換したものをPSのG-標準形, そのDigraphをGPSの標準形という。

〔命題5〕 GPSが標準形ならば. それは Connected Graph である。また. G-correctなPSのGPSは Start node から Halt node への path を有する。

〔命題6〕 PSが Weakly G-correctならば. PSは少なくとも1つ. 形式的Infinite Loop[*7]を含む.

証明. GPSを考える. $N_\ell = N_S - N_h$ とおく. 条件より $N_\ell \neq \phi$, $N_\ell \cap N_h = \phi$. だから Halt node $n_h \notin N_\ell$. 従って $\forall n_i (n_i \in N_\ell)$ に対して. Outdegree$(n_i) \geq 1$. かつ $n_i n_j \in A$ ならば $n_j \in N_\ell$. $\#N_\ell$: 有限 / a.E.D.

*7 Loop内の node から Loop外の node へ出る arc がない Loop.

〔命題7〕　PS′をPSのG-標準形とするとき．PS≡PS′．

(L,G)-correct**, (L,G)-標準形のPSはLabelに関しては．正しく望ましいものである。Controlのflowはps内で閉じており．形式的には．PS内の全てのpath上を通りうる。

** L-correct ∧ G-correct の意．

§6．プログラムの形式的正当性（その2）

6－1　諸定義および Notation

PSの Program Variable の集合を V_p とする．$V_p=\bigcup_{i=1}^{n}(V_i^{2d}\cup V_i^{2r})$．

〔定義21〕　$X\in V_p$ が $X\in V_i^{2d}$（$X\in V_i^{2r}$）であるとき．変数Xの S_i 上の defined (refered) occurrence という。また．まとめてXの S_i 上の occurrence といい．それぞれ X^d/S_i，X^r/S_i，X/S_i と表わす。

$$EDO(S_j,X,S_i)= (X^r/S_i\wedge X^d/S_j)\wedge\exists\nu(\nu=n_{k1}n_{k2}\cdots n_{k\ell}\wedge n_{k1}=n_j\wedge n_{k\ell}=n_i\wedge(1\le m<\ell\supset n_{km}n_{km+1}\in A)\wedge\neg\exists n_{kh}(h\neq 1\wedge X^d/S_{kh})) \qquad (1.6)$$

〔定義22〕　$EDO(S_j,X,S_i)$ ならば X^d/S_j は S_i の Effective defined occurrence である。このとき．S_j は S_i を支配するといい．$S_j\succ S_i$ と表わす。

〔定義23〕　次の Induction Definition によって定まる集合を S_i の Dominant Set $D(S_i)$ という。

1. $S_j\succ S_i$ ならば $S_j\in D(S_i)$
2. $S\in D(S_i)\wedge S'\succ S$ ならば $S'\in D(S_i)$

3. 1,2によって含まれるものだけが$D(S_i)$の要素である。

つぎに、ΣをPSのStatement Schemaの集合。Statement Schema S_iが、Predicate, Output, Halt Statement Schemaのとき、それぞれP_i, O_i, H_iと表わす。P_i, O_i, H_iは言語$\mathbb{L}$の$\mathbb{D}$によって、形式的に定められる。

〔定義24〕 Output Dominant Set Σ_o, Flow Dominant Set Σ_p, Halt Set Σ_Hを次のように定めるとき。

$$\Sigma_o=\bigcup_{O_i\in\Sigma}(D(O_i)\cup\{O_i\}),\quad \Sigma_p=\bigcup_{P_i\in\Sigma}(D(P_i)\cup\{P_i\}),\quad \Sigma_H=\bigcup_{H_i\in\Sigma}H_i$$

$\Sigma-(\Sigma_o\cup\Sigma_p\cup\Sigma_H)$の要素をr-Statement Schemaという。PSの全てのr-Statement Schemaを入で置換したものをS-標準形という。

〔命題8〕 PS′をPSのS-標準形とするとき、PS′≡PS.

〔定義25〕 PSの変数のdefined occurrenceの集合W^{2d}, Effective defined occurrenceの集合W^{2e}を次のように定める。

$$W^{2d}=\bigcup_{i=1}^{n}\{X^d/S_i \mid X\in V_i^{2d}\},$$

$$W^{2e}=\bigcup_{i=1}^{n}\{X^d/S_j \mid \exists S_i\, EDO(S_j,X,S_i)\}.$$

このとき、$W^{2d}-W^{2e}$の要素をr-occurrenceという。PSの全てのr-occurrenceを入で置換したものをV-標準形という。

〔命題9〕 PS′をPSのV-標準形とするとき、PS′≡PS.

明らかに、PSの(L, G, S, V)-標準形PS′は、PS′≡PSで、Loop内の一部以外では、冗長性のないものである。

6-2 Strongly Formal Correctness

〔定義26〕 GPSから次のプロセスによって作られるDigraphをdot付GPS. DGPSという。 $DGPS=\langle \mathcal{N}', A', D\rangle$, $\mathcal{N}'=\mathcal{N}$. $A'\geq A$, D : dot[*8]のfinite set.

1. $Indeg(n_i)\geq 1$ のnodeにin-dotを. $Outdeg(n_i)\geq 1$ のnodeへout-dotを与える。
2. in-dotからnodeへ, nodeからout-dotへarcを引く。
3. GPS中の n_i へのin-arcを n_i のin-dotへ, n_i からのout-arcをout-dotから書く.
4. $Indeg(d_i)=Outdeg(d_i)=1$ のdot d_i を消去し. in-arc, out-arcを1本にまとめる。

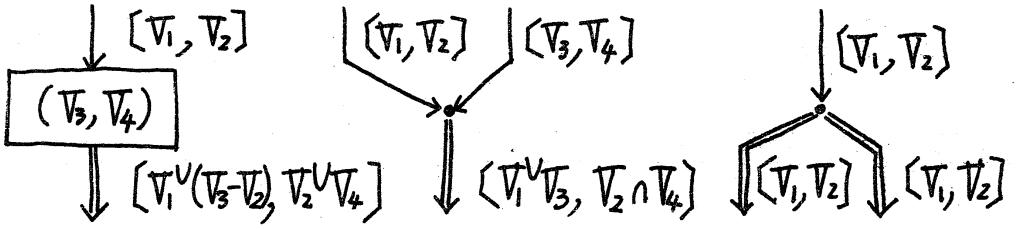
〔定義27〕 node n_i に対応する $S_i=\langle F_i, V_i^{1d}, V_i^{1r}\cup C_i^1, V_i^{2d}, V_i^{2r}\cup C_i^2\rangle$ によって定まるordered pair (V_i^{2r}, V_i^{2d}) を n_i のNode表現と呼ぶ。

〔定義28〕 次の規則によって定まるordered pairを. 対応するarcのArc表現と呼ぶ。

1. Start nodeのout-arc : $[\phi, \phi]$
2. Node表現 (V_3, V_4) のnodeのin-arcのArc表現を $[V_1, V_2]$ とするとき. そのout-arcのArc表現 : $[V_1\cup(V_3-V_2),\ V_2\cup V_4]$
3. Arc表現 $[V_1, V_2]$, $[V_3, V_4]$ の2つのarcのjoint arcのArc表現 : $[V_1\cup V_3,\ V_2\cap V_4]$

*8 対応するStatement Schemaを持たないnodeを区別してdotと呼ぶ。

4. disjoint arc の Arc 表現は. その dot の in-arc の Arc 表現と同じ. 規則の 2. 3. 4 をそれぞれ. fig. 1, 2, 3 に示す。



(fig. 1)　(fig. 2)　(fig. 3)

〔命題10〕　DGPS の全ての node と arc に. Node 表現. Arc 表現が定まる。

Node 表現 (V_1, V_2) の意味は. この Statement Schema 以前で. V_1 の値が定められていれば. この Statement Schema で V_2 の値が定まる. すなわち. この Statement Schema の全ての変数 V_i^2 の値が定まるということである。Arc 表現 $〔V_1, V_2〕$ の意味は. V_1 は Start node からこの arc へ到達しうる全ての path 上の全ての node (Statement Schema) の変数の値が. その値が定められると定義可能になるような最小集合, V_2 はその内のどの path を通って来ても. その値が定義されているような変数の最大集合を表わすということである。

〔定義29〕　DGPS において. Halt Statement Schema H_i (Output Statement Schema O_i) に対応する n_i の In-Arc 表現[*9] を $〔V_1, V_2〕$ とするとき. 全ての $H_i(O_i)$ の In-Arc 表現において $V_1 = \phi$ $(V^2 \supseteq V_i^{2n})$ であるとき. PS は I(O)-Close であるという。

*4　in-arc の Arc 表現の意味.

〔定義30〕　(L,G)-correctなPSが(I,O)-Closeであるとき、PSはStrongly Formal Correctであるという。

6－3　Weakly Formal Correctness

6－2 Strongly Formal Correctnessにおける Arc表現に関する〔定義28〕を次のように変更する。

〔定義31〕　z_iを$\langle \nabla_i, \nabla_i' \rangle$を表わすものとするとき、Arc表現を次のように定める。

1. Start nodeの out arc：　$\{\langle \phi, \phi \rangle\}$
2. Node表現(∇, ∇')のnodeのIn Arc表現を$\{z_1, z_2, \cdots, z_n\}$とするとき、そのOut Arc表現は$\{z_1', z_2', \cdots, z_n'\}$である。ただし、　$z_i' \equiv \langle \nabla_i \cup (\nabla - \nabla_i'),\ \nabla' \cup \nabla_i' \rangle$
3. Arc表現$\{z_1, z_2, \cdots, z_n\}$, $\{z_1', z_2', \cdots, z_m'\}$の2つのarcのjoint arcのArc表現は$\{z_1, z_2, \cdots, z_n, z_1', z_2', \cdots, z_m'\}$である。
4. disjoint arcのArc表現は、そのdotのin-arcのArc表現と同じ。

〔定義32〕　DGPSにおいて、Halt (Output) Statement Schema $H_i(O_i)$のIn Arc表現を$\{z_{i1}, z_{i2}, \cdots, z_{in}\}$, $z_{ij} \equiv \langle \nabla_{ij}, \nabla_{ij}' \rangle$とするとき、全ての$H_i(O_i)$のIn Arc表現において、$\nabla_{ij} = \phi$ $(\nabla_{ij} = \phi \wedge \nabla_{ij}' \supseteq \nabla_i^{2r})$なる$z_{ij}$が少なくとも1つ存在するとき、PSはWeakly I(O)-Closeであるという。

〔定義33〕　(L,G)-correctなPSがWeakly O-Closeであるとき、PSはWeakly Formal Correctであるという。

〔命題11〕　PSがStrongly or Weakly Formal Correctであるかどうかは決定可能である。

§7. 一般的正当性と形式的正当性の関係.

全てのプログラムの集合: Ω, 全ての入力に対して停止するクラス: $\Omega_{\forall F}$, 停止する入力が存在するクラス: $\Omega_{\exists F}$, 同一入力に対し同一のresult(undefineも含む)を得るクラス: Ω_Sとする。

〔命題12〕　Totally (partially) correctなプログラムのクラスは、$\Omega_{\forall F} \cap \Omega_S$ ($\Omega_{\exists F} \cap \Omega_S$)である。

全てのプログラム形の集合: Π, (L, Strongly(Weakly)-G)-correctなクラス: Π_{SG} (Π_{WG}), Strongly (Weakly) (I,O)-Closeなクラス: Π_{SC} (Π_{WC}), Strongly (Weakly) Formal Correctなクラス: Π_{SFC} (Π_{WFC})とする。

〔命題13〕　$\Pi_{SFC} = \Pi_{SC} \cap (\Pi_{SG} \cup \Pi_{WG})$, $\Pi_{WFC} = \Pi_{WC} \cap (\Pi_{SG} \cup \Pi_{WG})$.

Interpretationを I, interpretationを与えられたプログラム形PSを $\langle PS, I \rangle$ と表わす。

〔定理1〕　$PS \in \Pi_{SC}$ ならばかつそのときのみ $\forall I(\langle PS,I \rangle \in \Omega_S)$ である。また. $PS \in \Pi_{WC}$ ならばかつそのときのみ $\exists I(\langle PS,I \rangle \in \Omega_S)$ である。

〔定理2〕　$\forall I(\langle PS,I \rangle \in \Omega_{\forall F})$ ならば. $PS \in \Pi_{SG}$ である。また. $\exists I(\langle PS,I \rangle \in \Omega_{\exists F})$ ならば. $PS \in \Pi_{WG} \cup \Pi_{SG}$ である。

〔定理3〕　$\forall I(\langle PS,I \rangle \in \Omega_{\text{totally correct}})$ ならば. $PS \in \Pi_{SFC}$ である。また. $\exists I(\langle PS,I \rangle \in \Omega_{\text{patially correct}})$ ならば. $PS \in \Pi_{WFC}$

である。

§ 謝辞

日頃ご指導いただく、京大工学部 清野 武教授ならびに矢島脩三教授に深謝いたします。

参考文献

1. Church, A., (1956). "Introduction to Mathematical Logic", Princeton univ. press, vol. 1.
2. Church, A., (1941), "The Calculi of Lambda-Conversion", Princeton univ. press.
3. Bakker, J. W. (1969), "Semantics of Programming Languages" Advances in Inf. Sys. Sci. (Tou, J. T., ed), vol. 2.
4. Manna, Z. (1969), "The Correctness of Programs", J. of Computer and System Science 3., pp. 119–126
5. Engeler, E. ed (1971), "Symposium on Semantics of Algorithmic Languages", Lecture Notes in Math. 188 Springer-Verlag.
6. 椎沢輝昭 (1970), "計算理論の基礎", 統合図書.
7. 謝. 清野, (1971). "プログラムの正当性について", 電気関係学会関西支部連合大会.
8 Harary, F., (1969), "Graph Theory" Addison-wesley.